Name:_____

1. Explain the output of the program below.

Program

```
#include <iostream>
using namespace std;

int main() {
   char str1[11];
   char str2[11];
   cout << "Enter a string: ";
   cin >> str1;

   cout << "Enter another string: ";
   cin >> str2;

   cout << "str1 is \"" << str1
        << "\" and str2 is \""
        << str2 << "\"" << endl;

   return 0;
} // end main
```

Window's Visual Studio Output

Enter a string: 12345678901234567890123456789O
Enter another string: abcdefghijklmnopqrstuvwxyz
str1 is "uvwxyz" and str2 is "abcdefghijklmnopqrstuvwxyz"

Run-Time Error Message in Visual Studio Pop-up Window:

```
Run-Time Check Failure #2 - Stack around the variable 'str2' was corrupted.
```

Linux Output with the Same Program and Input

Enter a string: 12345678901234567890123456789O
Enter another string: abcdefghijklmnopqrstuvwxyz
str1 is "qrstuvwxyz" and str2 is "abcdefghijklmnopqrstuvwxyz"
Segmentation fault (core dumped)

2. In an expression with more than one operator, evaluate in this order:
   - (unary negation), in order, left to right
   * / % (remainder), in order, left to right
   + -, in order, left to right

Parentheses ( ) can be used to override the order of operations. Evaluate each of the following:

a) 6 + 3 * 5

b) (6 + 3) / 2

c) 4 + 2 * 3 - 5

d) 7 % 4 + 5 * 6

e) 1 + 2 + 3 % 2

Name:_____

3. Operations are performed between operands of the same type. If operands are not of the same type, C++ will automatically convert one operand to be the type of the other, called *type coercion*. The type coercion rules are:

1) `char, short, unsigned short` automatically promoted to `int`
2) When operating on values of different data types, the lower one is *promoted* to the type of the higher one according to the ranking:.

```
long double        (Highest)
double
float
unsigned long
long
unsigned int
int                (Lowest)
```

3) When using the assignment operator =, the type of expression on right will be converted to type of variable on left.

For the variables: `short myShort=2; int myInt=3; double myDouble=3;`
determine the **type and value** of the following expressions:

a) `myInt / myShort * myDouble`

b) `myDouble / myShort * myInt`

c) `myInt * myDouble / myShort`

4. You can explicitly convert a value to a specific type (called *casting*) by using

- C-Style cast: data type name in `()`: `cout << ch << " is " << (int)ch;`
- Prestandard C++ cast: value in `()`: `cout << ch << " is " << int(ch);`
- Both are still supported in C++, although `static_cast` is **preferred**:

```
cout << ch << " is " << static_cast<int>(ch);
```

Evaluate each of the following:

a) `float(2) + 5 / 2`

b) `2.0 + 5 / static_cast<float>(2)`

c) `(float)(2 + 5) / 2`

5. For the variables: `int myInt, myOtherInt; double myDouble, myOtherDouble;`

what value is assigned to each of the following variables?

a)
```
myDouble = myInt = myOtherDouble = 3.9;
```

b)
```
myDouble = myOtherDouble = myInt = 3.9;
```

c)
```
myDouble = myInt = myOtherInt = 7;
myInt += 3;
myOtherInt /= 3;
myDouble /= 3;
```

Name:_____

6. The header file `iomanip` contains the following manipulators to control the formatting of the output (and input) of numbers and strings.

Stream Manipulator	Description
`setw(n)`	Establishes a print field of at least size *n*
`fixed`	Displays floating-point numbers in fixed-point notation
`scientific`	Displays floating-point numbers in scientific notation
`showpoint`	Causes a decimal point and trailing zeroes to be displayed, even if there is no fractional part.
`setprecision(n)`	Sets the precision of floating-point numbers to *n*
`left`	Left justify the output
`right`	Right justify the output

Explain the output of the following program.

```
// Program to experiment with formatted output
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    cout <<"(default) with << setw(15) << right:" << setw(15) << right << endl;
    cout << "setprecision(3) << 1234.5678: " << setw(15) << right << setprecision(3) << 1234.5678 << endl;
    cout << "setprecision(4) << 1234.5678: " << setw(15) << right << setprecision(4) << 1234.5678 << endl;
    cout << "setprecision(6) << 1234.5678: " << setw(15) << right << setprecision(6) << 1234.5678 << endl;

    cout << "\ncout << fixed" << fixed << endl;
    cout << "setprecision(3) << 1234.5678: " << setw(15) << right << setprecision(3) << 1234.5678 << endl;
    cout << "setprecision(4) << 1234.5678: " << setw(15) << right << setprecision(4) << 1234.5678 << endl;
    cout << "setprecision(6) << 1234.5678: " << setw(15) << right << setprecision(6) << 1234.5678 << endl;

    cout << "\ncout << scientific" << scientific << endl;
    cout << "setprecision(3) << 1234.5678: " << setw(15) << right << setprecision(3) << 1234.5678 << endl;
    cout << "setprecision(4) << 1234.5678: " << setw(15) << right << setprecision(4) << 1234.5678 << endl;
    cout << "setprecision(6) << 1234.5678: " << setw(15) << right << setprecision(6) << 1234.5678 << endl;
    cout << "setprecision(6) << \"1234.5678\": " << setw(15) << right << setprecision(6) << "1234.5678" << endl;
} // end main
```

Output:

```
(default) with << setw(15) << right:
setprecision(3) << 1234.5678:       1.23e+003
setprecision(4) << 1234.5678:            1235
setprecision(6) << 1234.5678:         1234.57

cout << fixed
setprecision(3) << 1234.5678:        1234.568
setprecision(4) << 1234.5678:       1234.5678
setprecision(6) << 1234.5678:     1234.567800

cout << scientific
setprecision(3) << 1234.5678:      1.235e+003
setprecision(4) << 1234.5678:     1.2346e+003
setprecision(6) << 1234.5678:   1.234568e+003
setprecision(6) << "1234.5678":       1234.5678
```

Name:_____

7. Consider the user interaction of the program below.

Program	Window's Visual Studio Output

#include <iostream>
#include <iomanip>
using namespace std;
 	
int main() {
  const int SIZE = 11;
  char str1[SIZE];
  char str2[SIZE];
  cout << "Enter a string: ";
  cin >> setw(SIZE) >> str1;
 	
  cout << "Enter another string: ";
  cin.getline(str2,SIZE);
 	
  cout << "str1 is \"" << str1 << "\" and str2 is \""
  << str2 << "\"" << endl;
 	
  char ch;
  cout << endl << "Hit any key to continue...." << endl;
  cin.get(ch);
  return 0;
} // end main
	Enter a string: abcdefghijklmn
	Enter another string: str1 is "abcdefghij" and str2 is "klmn"
	Hit any key to continue....

a) Explain why the user was only able to input to enter the first string.

b) What would we need to do to fix the above program?

8. Complete the C++ program to calculate the hypotenuse of a right triangle using the Pythagorean theorem: $c = \sqrt{a^2 + b^2}$, where c is the length of the hypotenuse, and a and b are the other sides of the triangle.

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
     double a, b, c;

     cout << "Enter the length of sides a and b: ";
     cin >> a >> b;









     cout << "The length of the hypotenuse is " << c << endl;

     return 0;
} // end main
```

Name:_____

Operator Precedence and Associativity		
Operator	**Associativity**	**Usage(s)**
`::`	unary: left-to-right binary: right-to-left	
`() [] -> .`	left-to-right	parenthesis index object pointer/structure pointer dot operator
`++ -- - + ! ~ (type) * & sizeof`	right-to-left	increment and decrement unary negation and plus logical negation one's complement operator type cast indirection address-of/reference
`* / %`	left-to-right	multiply, division, remainder
`+ -`	left-to-right	addition and subtraction
`<< >>`	left-to-right	io: insertion and extraction bit-wise shift left and right
`< <= > >=`	left-to-right	comparisons for inequality
`== !=`	left-to-right	comparison for equality
`&`	left-to-right	bit-wise AND
`^`	left-to-right	bit-wise exclusive-OR
`|`	left-to-right	bit-wise OR
`&&`	left-to-right	logical AND
`||`	left-to-right	logical OR
`?:`	right-to-left	conditional
`= += -= *= /= %= &= ^= |= <<= >>=`	right-to-left	assignment
`,`	left-to-right	comma operator